$2.00

The Apple Shoppe MAGAZINE

VOLUME 1 No. 8 JOURNAL OF APPLE APPLICATIONS MAY/JUNE 1981

76/1206
COMPUTERAMA
$ 2.00

Contents

SUPER HIRES GRAPHICS EXPLAINED! 6

NEWS & VIEWS	The Apple Shoppe Changes Hands!	1
NEW PRODUCTS	Real Time Clock Pascal Animation Package Pascal 1.1 Update Kit	2
LETTERS TO THE EDITOR	Educational Technology Letter	4
GRAPHICS WORKSHOP	560 by 191 Plot Program by David E. Smith	6
UTILITIES WORKSHOP	Embedded Control Characters	13
PROGRAM REVIEWS	by Kirk Chase	16
PASCAL WORKSHOP	16 Sector Disk Copy by Allen Todd	17
STRING MAGIC	String Search Utility	22

Keep Ahead of Microcomputer Developments With the Professional's Choice

Interface Age is the most up-to-date source of microcomputer hardware and software advances. Whether you need to be informed for future purchases or to make comparisons, Interface Age should be #1 on your list.

- It has **more new product information** than any other small systems publication
- Indepth hardware and software reviews
- Software and hardware applications
- Programming
- Robotics
- Book reviews
- Business applications
- Educational applications
- Latest technologies
- Tutorials

Take advantage of this no-risk trial subscription offer. If for any reason you are not completely delighted with the first issue, we will refund your payment in full. Order your subscription to INTERFACE AGE now with this convenient coupon.

NO-RISK TRIAL SUBSCRIPTION ORDER

12 issues for $18.00. That's a 40% savings off the single copy price. And if you are not delighted with the first issue, your payment will be refunded in full.

MAIL TODAY

- ☐ **One Year (12 issues) $18 (U.S. only)**
- ☐ **Two Years (24 issues) $30 (U.S. only)**
- ☐ **Canada/Mexico One Year $20**
- ☐ **Foreign Surface Mail One Year $35**
- ☐ **Foreign Air Mail One Year $50**

☐ **Check enclosed**
Must be in U.S. funds drawn on U.S. bank

Payment must accompany this order.

☐ VISA ☐ master charge ☐ AMERICAN EXPRESS Card

Card No. _____ Exp. Date _____

Signature _____

Name _____

Street _____

City _____ State _____ Zip _____

Make check or money order payable to: **INTERFACE AGE Magazine**
P.O. Box 1234, Dept. AS1, Cerritos, CA 90701

Allow 6-8 weeks for delivery.

Dear Subscriber,

I apologize for the long delay in the publishing of Vol. 1 No. 8 (yes, No. 7 was the last edition!) I would like also to thank you for your patience and the support you have given Dave Smith during this period. Dave will continue his fine tutorial style of writing and helping you learn more about your APPLE as contributing editor. I will concentrate on the publishing aspect (subscriptions, advertising and production) where I have many years of experience.

All subscribers will receive the number of issues to which they subscribed. We will publish eight issues a year.

This issue should be very helpful to you, especially the Graphics Workshop and String Magic.

I would also like to bring your attention to the Personal Computing To Aid The Handicapped Competition. This is a program that well deserves your attention and if you have developed any expertise in this area please contact John Hopkins University.

Those of you who like to write articles, please let me know. Contributors are always welcome. It would also be helpful to hear from you about what areas we should be looking into and what your problems and concerns are in the fascinating personal computer area.

I look forward to a long friendship and a mutually prosperous venture for all of us.

Sincerely,

Joseph L. Rollinson

Joseph L. Rollinson,
Publisher

The Apple Shoppe Magazine is published eight times a year for the purpose of educating and promoting practical use of Apple Computers. Address all editorial, advertising and subscription inquiries to THE APPLE SHOPPE MAGAZINE, P.O. Box 22026, San Diego, CA 92122. (714) 270-4521

SUBSCRIPTIONS: $12.00 per year for Continental U.S.A. and $24.00 per year outside the U.S. Overseas air mail rates are $36.00 per year U.S. funds. Payment must accompany subscription request.

DEALER SALES: THE APPLE SHOPPE MAGAZINE is sold world wide through computer stores. Prices will be quoted upon request. Phone or write to above address.

BACK ISSUES: Only photocopy issues are now available at a cost of $4.50 each. Write to above address.

ADVERTISING RATES: Contact above address for current rate card. New product information is solicited.

Entire contents copyright ©1981 by The Apple Shoppe Magazine. All rights reserved. Nothing may be printed in whole or part without written permission from the publisher.

Apple is a registered trademark of Apple Computer Inc.

News & Views

APPLE SHOPPE CHANGES HANDS!

For all of our patient (and not so patient) subscribers, we apologize for the silence these last six months. The success of the Apple Shoppe has been so overwelming that we have been forced to sell the magazine to a professional publisher in order that it might grow and expand into a regular monthly magazine. So this is the last issue I will be acting as Editor and Publisher. With this issue my involvement with the Apple Shoppe will come to an end, although I will continue to submit articles that I hope will communicate the Apple secrets in a way that each of you can gain something new and useful in your Apple pursuits. I know you all will be glad to see the magazine continue and on a more regular basis so please support our new publisher and give them time to put together a real class act for all of us. Laura and I appreciate the help and support all of you have given our "mom and pop" operation and hope that support will continue with the new management. My first assignment as "ex-editor" and "ex-publisher" is to be a reporter and cover the Sixth West Coast Computer Faire, so look forward to the next issue for full faire coverage.

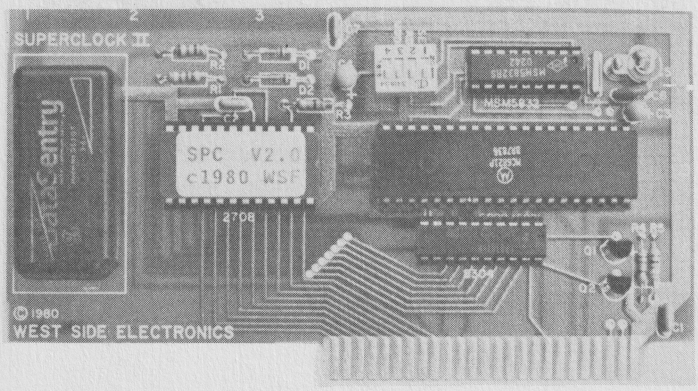
Regards,

David & Laura Smith
607 Twilight
Placentia, Ca. 92670

REAL TIME CLOCK

SUPERCLOCK II is a new real time clock for the Apple II computer. This compact board plugs into any slot of the computer and provides time and calendar information as well as advanced interrupt capabilities. Other features include: timing from milliseconds to years, control firmware in PROM, write protect switch, and 12/24 hour format. The SUPERCLOCK II is easily read from BASIC, Applesoft, and PASCAL. Of special interest is an Apple Clock emulation mode which allows the SUPERCLOCK to work with existing software written for the Mountain Hardware clock.

Up to four software controlled interrupts can be used to allow foreground/background programming or other real time activities. Pascal users will especially appreciate the automatic updating of the Filer and each disk that is booted. Of course the clock can also be used for printing headings, data logging, timing events, etc. The rechargable battery will typically last about 10 years.



Taking a systems approach, the company also offers several utility programs to take full advantage of the SUPERCLOCK II. For example, Auto-dater is a small DOS routine that automatically maintains a date for each file in the catalog. The SUPERCLOCK II is available from West Side Electronics, po box 636, Chatsworth, CA. 91311. PRICE IS $99.95.

PASCAL ANIMATION PACKAGE

APPLE has anounced a new distribution method for user developed software that is raising a lot of controversy. The program is called Special Delivery software that is not under normal Apple guarantees, but is distributed by Apple as a service to get good software out in the hands of the users without the normal review cycle a large corporation must go through to generate manuals etc.

One program released that looks very intriging is the Pascal Animation Package. According to the catalog, the software package allows you to use animation with Pascal and high resolution graphics without the time consuming bother of detailed assembly language. The technique appears to be the trick to mapping text on the text page to the hires page as developed by Bob Bishop in his classic t.v. dancing man program. Whether this would provide the same speed as Bishop's program remains to be seen. The main advantage to this program is that the source code is provided which means the user can gain the technical understanding of how to create animation. That fact alone makes the $75 price worth it. Included in the package are routines for using the Apple graphics tablet. All in all if the program lives up to it's billing, it should be a good

investment. As with all software, try to see a demonstration before you order.

Another interesting animation package in the same series is the Pilot Animation package. The user friendly design of this program appears the best I've ever seen. It comes very close to the ease of smalltalk in creating animated shapes such as the famous Maxwell. Requires the new Pilot author system and two disk drives and also sells for $75. A must for schools or school children whose parents have Apples.

PASCAL 1.1 UPDATE KIT

Apple has anounced a new Pascal version 1.1 update kit for $60. This update corrects many bugs in the origional package and provides two beautifully big colorful manuals to replace the horrible white elephant manual we've all struggled with. Some of the features include chaining programs with 80 character strings passed between them. Does that mean no global variables between chain segments?

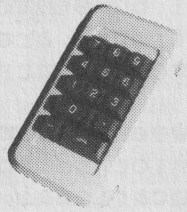
The ability to create an exec file will be most handy. This will reduce some of the big system nonsense that is required to use pascal by allowing requent proceedures to be placed in an exec. The I/O routines have been speeded up, whatever that means. I didn't notice they were slow. Swapping has been increased which is definitely a big help for large programs. The number of segments has been increased to 16, which is vital again for breaking up large programs.

Apple has tried and failed again to deal with the upper/lower case problem. In this version they have upper and lower case option as inverse video. Why don't they just acknowledge the existance of the Dan Paymer board and allow true lower case for those who have it which is most everybody? Now does anyone have a copy of the new BIOS routines so we can patch version 1.1 for the Dan Paymer board and line feed fix??

Apple Fortran has been reported to have a number of bugs which Apple has acknowledged. Look for Apple to release an update for Fortran in the near future also. The update won't have greater percision as that problem is in the P-code operating system. But SoftTech has announced a new P-code system that fixes this problem. If Apple buys it and distributes it, we might yet get a useful Fortran out of the Pascal system.

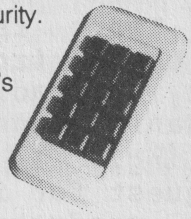
IMPROVE Your Data Entry!

With ABT Apple Peripherals*



KeyPad™ Used for entering numeric data, it is essential to business applications. It features an accountants keyboard layout and permits a relaxed arm position.

BarWand™ Compatible with U.P.C., Paperbyte**, Label Code and others. It sells Point-of-Sale inventory systems, but is also useful in libraries, factories and for security.



SoftKey™ A great programmer's aid, this features single key string entry and also customized key functions.

Available from your local Apple Dealer*

ADVANCED BUSINESS TECHNOLOGY, INC.

12333 Saratoga Sunnyvale Road, Saratoga, California 95070 408/446-2013

*Trademark of APPLE COMPUTER, INC. **Trademark of McGRAW-

Letters to the Editor

Dear Editor,

Educational Technology, now in its 20th year of publication, is in the process of expanding considerably its monthly coverage of the following three fields, due to reader requests:

1. Computers in education and training--hardware and software.

2. Audio-visual communication and AV products, including both older and new media.

3. Training in business and industry.

We are interested in receiving from you all appropriate information regarding products and services your firm is offering in these three fields, or in any one of these fields. Our readers, in all 50 states and in more than 80 nations, are eager to learn of new developments in education.

Lawrence Lipsitz
Senior Editor
Educational Technology Publications
140 Sylvan Ave.
Englewood Cliffs, N.J. 07632

Dear Mr. Lipsitz,

There is a revolution brewing on the horizon for education and the traditional educational organizations don't even see it coming. In response to your request for information regarding our products, we are trying to make that revolution happen as soon as possible. We publish The Apple Shoppe, the Journal of Apple Computer Applications. I have enclosed a sample copy of the latest issue. The arrival of the personal computer will truly re-vamp education in this country in the next ten years and we are determined to make the know-how available to our subscribers so that they can help make it happen. Our publication is highly oriented toward education, not in the traditional sense, but as it will be in the future. To that end our publication tries to help our readers get more out of the Apple and learn how to apply it to specific applicatons.

The organization that should be right on top of this movement towards personal learning with home computers is the National Council of Teachers of Mathematics, but they just bury their heads in the sand refusing to open the eyes of their members to the spread of these machines among their students, seemingly paralyzed over how to handle the personal computer in the classroom. How can a student get excited about taking four weeks to learn three different methods of solving the quadratic formula when in one day he wrote a short BASIC program to solve all quadratic equations on this Apple computer? How can a history student get excited about a three week lecture series on the civil war when each night he goes home and participates in an involved simulation of the march of General Sherman through the south on his home computer? How can an art appreciation student get excited about a dull lecture on water colors when at night he goes home and "paints" 21 color animations using the computer and a graphics tablet on his Apple?

Our publication and the educational software we are developing is aimed directly at bringing knowledge of these tools to our readers and helping develop the software tools that will make the personal computer a true exploration machine for our youth.

David E. Smith
Contributing Editor

MR. RAINBOW SAYS...

BLAST-OFF

WITH OUR SPECIAL CATALOG FEATURE...

A STELLAR TREK

OUR HI-RES COLOR VERSION OF THE CLASSIC STARTREK GAME WITH 22 PAGES OF DOCUMENTATION NEEDS 48K APPLESOFT ROM DISK.... $24.95

WRITE OR CALL TODAY FOR YOUR FREE CATALOG (OVER 100 PAGES - INCLUDES PRODUCTS FROM EVERY MAJOR MANUFACTURER.)

FOR INNER STELLAR DELIVERY (BARRING KLINGONS) ADD $2.00 US/ $10.00 FOREIGN FOR SHIPPING AND 6% TAX IN CALIFORNIA

apple computer inc.

OPEN TUES. - SUN.
VISA / MASTERCHARGE WELCOME

RAINBOW COMPUTING INC.

GARDEN PLAZA SHOPPING CENTER
9719 RESEDA BOULEVARD DEPT. 1AS
NORTHRIDGE, CALIFORNIA 91324
PHONE: (213) 349-0300

THE WIZARD AND THE PRINCESS
HI-RES ADVENTURE #2

Only ON-LINE SYSTEMS could deliver a HI-RES ADVENTURE game on such an epic scale. In this adventure you find you must do battle against an evil wizard in order to save the life of the princess. To find the wizard and his castle you must first cross deserts, oceans, mountains, travel to an island and encounter many strange beasts. You will be forced to learn magic, navigate at sea and dig for treasure. This game should provide months of adventure.

- HUNDREDS OF HI-RES PICTURES (looks great on b/w and color televisions)
- FULL 21-COLOR!! HI-RES GRAPHICS (each room a work of art)
- YOUR GAME MAY BE SAVED FOR LATER CONTINUANCE
- RUNS ON BOTH 48K APPLE II AND APPLE II PLUS
- BY FAR THE MOST AMBITIOUS GRAPHIC GAME EVER WRITTEN FOR THE APPLE!!

Hi-Res Adventure #2 is available now at your local computer store and requires a disk drive. To order directly send $32.95 to:

On-Line Systems
36575 Mudge Ranch Road
Coarsegold, CA 93614
209-683-6858

VISA, MST CHG, COD, CHECK ACCEPTED

Look for Hi-Res Football coming soon

SAVE 25%
over single copy rate.

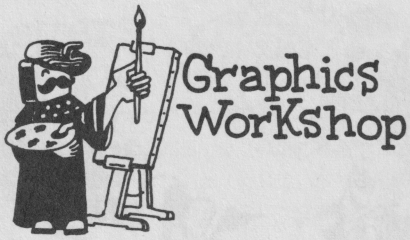
Subscribe to your own issue of THE APPLE SHOPPE MAGAZINE at $12.00/year. ($24.00 outside U.S./$36.00 overseas airmail.)

Name _____

Address _____

City _____ State _____ Zip _____

Send to: THE APPLE SHOPPE MAGAZINE
P.O. Box 22026
San Diego, CA 92122



SUPER HIRES GRAPHICS

Recently, some new information has come to light concerning the Apple's hires graphics. It seems that the Apple is capable of not only 279 X 191 resolution but of 559 X 191 resolution in black and white. And this doubly dense graphics mode is used in the Apple III. What has not been known until Bob Bishop published a short article on it in the fall 1980 issue of The Apple Orchard, is that this 559 by 191 black and white resolution is available for the Apple II as well. In this month's Graphics Workshop we present a short program in Applesoft that illustrates how to use this new hires mode for plotting more accurate functions than has been available in the past.

HOW WE GET 560 BY 192 POINTS

How is it that we can double the horizontal resolution of the Apple screen without hardware modifications? The key lies in the fact that the various color modes cause a shift in the physical position of the dot on the screen that on a black and white monitor appear to double the effective resolution of the screen. Through the use of software, we can use this color shift to create effectively twice the normal 279 dot resolution as long as we use a black and white monitor. As Bob Bishop points out in his article cited above, the Apple really has two distinct modes: 140 by 192 dots in six colors and 560 by 192 dots in black and white. Attempting to combine the two modes into a universal "monitor independent" mode created the artificial 280 by 192 dot mode in six colors with restrictions. If color is important, the 140 by 192 mode is used as in Apple Pascal to give true color lines. Where color is not important, the 560 by 192 mode can give good sharp detail in black and white. But unfortunately most Apple products have been designed for the mythical 280 by 192 as in the Apple Plot program. To further stimulate the use of this new 560 by 192 mode, Compu-Tutor is designing a second generation Apple Plot program featuring this new resolution and offering the same features of the currently popular Apple Plot program including Visicalc interfacing. But this new program will also feature a general purpose functional plotter to allow curve fitting to Visicalc data files, something not offered with Apple Plot. This program uses the techniques discussed in this month's workshop and when completed will be availalbe retail for $50. Let us know if you would lik advance information on this program as it becomes available.

HOW HIRES REALLY WORKS

Many articles have been written attempting to explain how the hi-res graphics mode works. Some have used look-up tables where the computer looks up the physical address of the desired row in a table of addresses that have been previously computed. Some use assembly language routines to compute the correct row address and then the proper byte in that row. But none of these articles have really explained to my satisfaction how it works. After much thought and study I attempt yet another explanation. The following little program fragment really reveals it all much better than a lot of prose.

```
]LIST
5 BASE=8192:HGR:HOME
10  FOR J = 0 TO 191
20  FOR I = 0 TO 39
30 TARGT = BASE + I + ROW
40  POKE TARGT,255
50  NEXT I
60 ROW = ROW + 40
70  IF (J + 1) / 3 =  INT ((J + 1
    ) / 3) THEN ROW = ROW + 8
80  NEXT J

]
```

Run this program and observe carefully what happens. Line by line the entire screen will fill up but not in sequential order. The key is in line 30 where we construct the address in memory of the next byte to poke. To the base address of the start of the hires buffer at $4000 (16384), we add I bytes in the current row and ROW, the number of bytes in each previous row. So we use a double loop to increment 40 bytes per row from 0 to 39 for all 192 rows from 0 to 191. At each byte we poke $FF (255) so that each bit is turned on, producing a short white line. This fills up the entire screen.

This seems simple enough except there is one little problem. Multiply 40 bytes per row times 192 rows and what do you get? You get 7,680 which is little less than 8,192 (8K) you expect. What happened to the extra bytes? Here is where the real mystery is about hires graphics. It seems that the physical memory is not continuously mapped to the display screen. Every 3 rows there are eight bytes that are NOT mapped to the screen. Therefor if you poke every byte in the hires buffer continuously using 40 bytes per row for 192 rows, you won't cover the screen! This can be verified by re-running the above program after deleting line 70, which checks each third row and bumps the ROW counter by 8 bytes to skip over the unused portions of the display buffer. What line 70 does is a modulous 3 function. If (J+1)/3 = INT ((J+1)/3) then J+1 is exactly divisable by three and hence it's time to bump the row counter by 8 more unused bytes. So what we have is a whole lot of extra register space embedded in 8 byte chunks in the hires display buffers.

560 DOT LOCATIONS

Now that we understand the row mapping in simple BASIC, lets see if we can understand the color mapping in each byte in the row that gives us this extra resolution in terms of simple poke statements. In the example program below look carefully at the two FOR-NEXT loops on I and J. The I loop is incrementing across the row of 40 bytes one byte at a time in lines 10 and 20. Then for each byte, we execute an inner FOR-NEXT loop on J that goes from bit 0 to bit 6 in each byte. The key to the 560 dot locations is in selectively turning on and off the 8th color bit as each of the 6 bits 0 to 6 is turned off or on. Each bit has the color bit first turned off and then turned on as shown in lines 55 (off) and 60 (on). The result is that each hires bit 0 to 6 is really plotted in two postions on the screen depending on the state of the high order bit of that byte. To prove that there are really 560 distinct dot positions, run this program and watch the dot move slowly across the screen while a counter counts 560 postions. Again the key is lines 35 to 80 that decide when to turn the hires bit on and off while turning on each succeeding hires bit in the byte.

Once we figure out which bit to turn on to get the proper dot on the screen, we need some way to turn that bit on correctly. The subroutine in lines 100 to 160 in example 2 below set or clear the proper bits so as to make or erase a dot on the screen without disturbing any of the other dots. This subroutine in turn calls a

short machine language routine also shown below that turns the bits on or off depending on the variable Z. Hex location 300 is the start of the set bit routine, while hex 30A is the start of the clear bit routine. We pass the bit position from variable Y to location $325 and the byte from variable X to location $324. The desired bit in the desired byte is set or cleared by the machine language routine using ORA instruction to set it, and the AND instruction to clear it.

Finally a short stall routine allows us to carefully watch our dot move across the screen so we really believe there is 560 distinct dot positions. You should see a small dot move slowly while the counter counts each position from 0 to 559.

```
]LIST

5 BASE = 8192: HGR : HOME
10  FOR I = 0 TO 39
20 TARGT = BASE + I
25  FOR J = 0 TO 6
30 X =  PEEK (TARGT)
35 Y = J:Z = 1: GOSUB 100
40  IF J <  > 0 THEN Y = J - 1:Z =
     0: GOSUB 100
45  IF (J = 0 AND I <  > 0) THEN
     Y = 6:Z = 0:W = X:X =  PEEK
     (TARGT - 1): GOSUB 100: POKE
     TARGT - 1,X:X = W
50 Y = 7:Z = 0: GOSUB 100
55  POKE TARGT,X: GOSUB 200
56 CT = CT + 1: VTAB (22): HTAB (
     1): PRINT "      ";: HTAB (1
     ): PRINT CT
60 Y = 7:Z = 1: GOSUB 100
65  POKE TARGT,X: GOSUB 200
66 CT = CT + 1: VTAB (22): HTAB (
     1): PRINT "      ";: HTAB (1
     ): PRINT CT
80  NEXT J
85  NEXT I
90  PRINT  CHR$ (7)
95  END
100  REM  BIT SUBROUTINE (X,Y,Z)
105  REM  X=CONTENTS OF TARGET BY
     TE
110  REM  Y=BIT POSITION 0-7 R-L
115  REM  Z=1 TO SET,0 TO CLEAR
120  POKE 804,X
130  POKE 805,2 ^ Y
140  IF Z = 1 THEN  CALL 768
150  IF Z = 0 THEN  CALL 778
155 X =  PEEK (804)
160  RETURN
200  REM  STALL ROUTINE
210  FOR K = 1 TO 100: NEXT
215  RETURN
]

]CALL-151

*300L

0300-   AD 24 03    LDA   $0324
0303-   0D 25 03    ORA   $0325
0306-   8D 24 03    STA   $0324
0309-   60          RTS
030A-   08          PHP
030B-   48          PHA
030C-   18          CLC
030D-   A9 00       LDA   #$00
030F-   ED 25 03    SBC   $0325
0312-   8D 25 03    STA   $0325
0315-   AD 24 03    LDA   $0324
0318-   2D 25 03    AND   $0325
031B-   8D 24 03    STA   $0324
031E-   68          PLA
031F-   28          PLP
0320-   60          RTS
0321-   00          BRK
0322-   00          BRK
0323-   00          BRK
0324-   00          BRK
*
```

NOW THE FINAL PLOT PROGRAM

Now that we have mastered hires, we can return our attention to BASIC and our plotting application. The program here will plot single line functions using this new 560 dot mode. Let's see how it works.

In the section of code below we set up the parameters used in the plotting of our function. The function itself is in line 2050 in the format "y = f(x)". Change this line to plot different functions. Each function must be a single valued real function of a single real variable. The physical plot limits of the screen are defined by A and B. Typing return at any

input causes default values to be used so the first time the program is run you can simply default everything to get a feel for how the program operates. The radian option is not used in this version. Both the minimum and maximum x and y values may be specified or they may default to insure the function will fit on the screen. This initial setting up fills lines 10 to 85 as shown below.

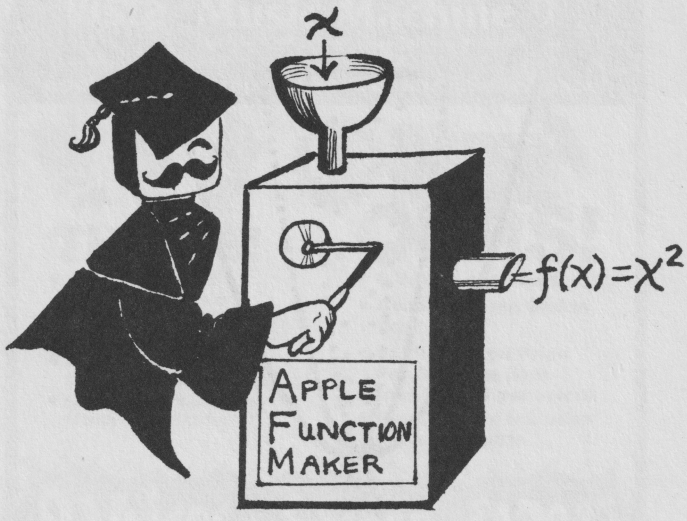
PLOT PROGRAM - INIT SECTION

```
10  TEXT : HOME : VTAB 3: HTAB 10
     : INVERSE : PRINT "SUPER-HIR
    ES FUNCTION PLOTTER": NORMAL

20  VTAB 5:M0 = (191 / 559):S0 =
    191 / 2:TRIG = 0
30  INVERSE : PRINT "PLOT LIMITS:
    ": NORMAL : PRINT
35  INPUT "START AND END VALUE (-
    95.5 TO 95.5)?";A$,B$
36  IF A$ = "" OR B$ = "" THEN A =
     - 100:B = 100: GOTO 40
37 A =  VAL (A$):B =  VAL (B$)
40  PRINT : INVERSE : PRINT "TRIG
    OMETRIC:": NORMAL : PRINT
45  INPUT "CONVERT TO RADIANS (Y
    OR RET)?";KEY$
50  IF KEY$ = "Y" THEN A = A * 3.
    14159 / 180:B = B * 3.141591
    80:TRIG = 1
60  PRINT : INVERSE : PRINT "SCAL
    ING:": NORMAL : PRINT
65  INPUT "X-MIN, X-MAX?";XN$,XM$

70  IF XN$ = "" OR XM$ = "" THEN
    XN =  - 100:XM = 100: GOTO 7
    5
73 XN =  VAL (XN$):XM =  VAL (XM$
    )
75 X = XN: GOSUB 2050:Y0 = Y:X =
    XM: GOSUB 2050:Y1 = Y
77  PRINT : PRINT "Y-MIN= ";Y0;"
        Y-MAX= ";Y1
80  PRINT : INPUT "Y-MIN, Y-MAX O
    R RET?";YN$,YM$
82  IF YN$ = "" OR YM$ = "" THEN
    YN = Y0:YM = Y1: GOTO 85
83 YN =  VAL (YN$):YM =  VAL (YM$
    )
85  REM  CONTINUE

]
```



Next comes the main portion of the program. Here we have the option of clearing the screen for a new plot, or plotting on top of whatever is there. This is useful for plotting a family of hyperbolas for example, to see how a change in one parameter affects the overall plot of the function. If you wish to save your plots, page 2 of the hires buffer is used so you should issue a "BSAVE" command for the second hires buffer located from hex 4000 to hex 5FFF.

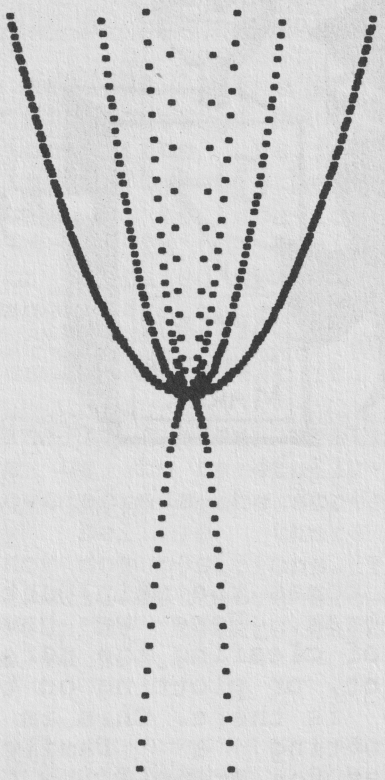
```
]

]LIST 86,199

100  REM
          BEGIN MAIN;

105  PRINT : INVERSE : PRINT "SIN
     GLE PLOT OR FAMILY:": NORMAL
     : PRINT
110  INPUT "SAME DISPLAY (Y OR RE
     T)? ";KEY$
120 SAME = 0: IF KEY$ = "Y" THEN
     SAME = 1
130  IF SAME = 0 THEN  HGR2
140  POKE  - 16304,0: POKE  - 163
     02,0: POKE  - 16299,0: POKE
      - 16297,0

]
```



The actual plotting begins in the next section. A counter I is used to count all 560 possible horizontal locations accross the screen. A flag called NOGO is used to tell us if a dot is required at each horizontal location. If so we call our super hires subroutine at line 1000 to place the dot there. Otherwise we move to the next possible location with a NEXT I command in line 295 until the plot is finished. Some scaling is required on x and y to offset the fact that 560 by 191 does not produce a square array on the monitor and to offset the origin from the upper left hand corner to the middle of the screen as would normally be expected in a cartesian x-y coordinate system. This is done in lines 220 and 240 using M0 and S0 to scale the graph. Note that the super hires subroutine requires the x and y coordinates to be integers. To do this we pass x and y as x% and y% before calling line 1000.

```
]

]LIST 200,999

200  REM

          560 POINT MODE

210  FOR I = 0 TO 559
220 X = M0 * (I - 559 / 2)
230  GOSUB 2000: REM  FUNCTION
240 Y = S0 - Y
290  IF NOGO = 0 THEN X% = I:Y% =
     Y: GOSUB 1000: REM  PLOT
295  NEXT I
299  VTAB 22: PRINT "";: END


]
```

SUPER HIRES SUBROUTINE

The heart of the program is the super hires plot subroutine. This subroutine takes an x% and y% in the range of 0 to 559 or 0 to 191 and plots a black and white dot at the proper location. This is done by using the HCOLOR function to shift the dot depending on the position. It is this shifting that gives us the extra resolution. This is the same as setting the high order color bit in each byte. The color factor divides each base byte into four color sections, each providing 3.5 dots. Hence each base byte yields 14 distinct dots. Since there are 40 bytes to a line, this gives 40 times 14 or 560 dots per row. Now how can their be 3.5 dots per section you ask? Recall that in the 280 by 191 mode that one bit from the odd bytes matches up with one bit from the adjacent even byte to produce one dot of color, so that in the 280 by 191 mode we have 14 bits out of two bytes to produce 7 distinct dots on the screen, or 3.5 dots per hires byte in that mode. Now by shifting the color bit on every other combination, we double those 7 distinct dots to 14 distinct dots per hires byte. Recall that the

Applesoft BASIC is set up in the 280 mode and hence when using HPLOT as shown below, we have to trick Applesoft into the 560 mode without him knowing about it!

```
1000  REM
          SUPER-HIRES PLOT SUB.

1005  REM  X% FROM 0 TO 559
1006  REM  Y% FROM 0 TO 191
1030 XX% = X% / 4:M% = X% - 4 * X
     X%
1040  IF M% = 0 THEN  HCOLOR= 2
1050  IF M% = 1 THEN  HCOLOR= 6
1060  IF M% = 2 THEN  HCOLOR= 1
1070  IF M% = 3 THEN  HCOLOR= 5
1080  HPLOT X% / 2,Y%
1090  RETURN : REM

2000  REM
          DEFINE FUNCTION & LIMI
     TS

2005 NOGO = 0
2010  IF X < A OR X > B THEN NOGO
      = 1: RETURN
2050 Y = X
2060  IF Y < ( - 191 / 2) OR Y >
     (191 / 2) THEN NOGO = 1: RETURN

2090  RETURN

]
```

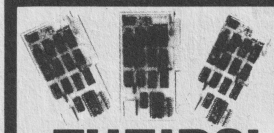
We hope you have enjoyed this look into hires graphics. This information is not generally known to our knowledge. Still to come is translating what we have learned into machine language so we can get speed into our routines, and learning mapping schemes that produce the effect of 21 colors instead of 6. So we have lots to learn and hope you'll stay with us in the GRAPHICS WORKSHOP.

BACK ISSUES

Back issues are available but only in photocopy, since all original publications have been sold out. Each edition is $4.50 and can be ordered from THE APPLE SHOPPE MAGAZINE, P.O. Box 22026, San Diego, CA 92122.

Vista Music Machine 9

Apple Users Compare! Save Dollars! Save Space!



AVAILABLE NOW from stock!

THEIRS!

- *3 Voices per Board (some others have only 1 voice per board)*
- *Consumes 3 Slots for 9 Voices*
- *$269.00 List Price. Requires $807.00 for Complete System*
- *One Board Produces Only One Monophonic Channel*

OURS!

- **9 Voices per Board**
- **Uses Only One Slot in Your Apple**
- **$129.95 Total Price For 9 Voices (less than $15.00 per voice)**
- **Single Board Includes Stereo Outputs**

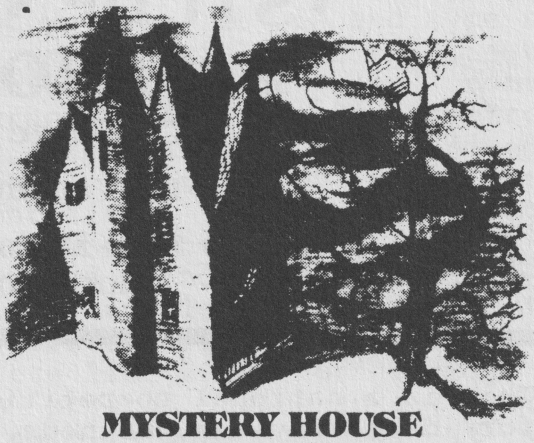
9 Voices ★ Stereo ★ $129.95

The Vista Music Machine 9™ utilizes new state-of-the-art LSI Sound Generation devices. Price includes demonstration diskette and a conversion program allowing the Music Machine to be used with ALF™ music programs (not supplied). Installation instructions are also included. The Music Machine 9™ is the lowest priced music board on the market and it is completely assembled and tested, ready to plug into your Apple. Available from stock.

Dealer inquiries invited.

Vista Computer Company
1401 Borchard St., Santa Ana, CA 92705
(714) 953-0523

Call Toll-Free
800-854-8017
International Inquires Invited



MYSTERY HOUSE
HI-RES ADVENTURE #1

Your APPLE computer becomes your eyes and ears as you enter a spooky old mansion in search of treasure. You are in complete control as you open cabinets, smash walls etc. Danger is ever present as you find your co-adventurers being murdered one by one. Can you find the killer before the killer finds you?

- OVER A HUNDRED HI-RES PICTURES
- YOUR GAME MAY BE SAVED FOR LATER CONTINUANCE
- RUNS ON BOTH 48K APPLE II AND APPLE II PLUS

Hi-Res Adventure #1 is available now at your local computer store and requires a disk drive. To order directly send $24.95 to:

On-Line Systems
36575 Mudge Ranch Road
Coarsegold, CA 93614
209-683-6858
VISA, MST CHG, COD, CHECK ACCEPTED

Look for Hi-Res Football coming soon

PERSONAL COMPUTING TO AID THE HANDICAPPED

THE JOHNS HOPKINS FIRST NATIONAL SEARCH

HEARING & LANGUAGE * MENTAL RETARDATION * LEARNING DISABILITIES * VISION
NEUROMUSCULAR/NEUROLOGICAL DISORDERS * LIMITATIONS OF MOVEMENT

$10,000 GRAND PRIZE

More than 100 other Awards:

- **Cash Prizes**
- **Personal Computers**
- **Certificates of Achievement**

• STUDENTS • AMATEURS • PROFESSIONALS

- **Challenge Your Creativity**
- **Put Your Innovative Ideas to Work**
- **Help Millions of Handicapped Persons Live More Productive and Rewarding Lives**

This is a national competition for ideas, devices, methods, and computer programs to help handicapped people overcome difficulties in learning, working, and successfully adapting to home and community settings.

This program is made possible by grants from THE NATIONAL SCIENCE FOUNDATION and RADIO SHACK, a division of TANDY CORPORATION.

To obtain additional information including a descriptive flyer and contest application, write to:

PERSONAL COMPUTING TO AID THE HANDICAPPED
THE JOHNS HOPKINS UNIVERSITY
P.O. BOX 670
LAUREL, MARYLAND 20810

Contestants will have until June 30, 1981 to prepare and submit their entries.

EMBEDED CONTROL CHARACTERS

Have you ever made the mistake of typing a control character into the name of a program or file you were saving and then found you had no way to retrieve that program or file? If so, then you will appreciate this utility that prints any embedded control characters in the catalog so that the name of the file can be specified properly. Unless this is done, there is no way to retrieve that information since the DOS will continually report "file not found" even though it appears you are typing the name of the file correctly. The reason for this is that control characters are not printed, but they are tested when checking for the name of the file. If you have a program you cannot seem to load even though you type what appears to be the correct name of the program, try running this utility and see if the name of the file doesn't contain some hidden control characters.

There are several important concepts contained in this program and all of them are discussed in the June 1979 issue of CALL-A.P.P.L.E. which you may wish to refer to on pages 5 -9. That issue contained a number of short programs dealing with the problem of using the monitor commands from a BASIC program or linking assembly language programs to BASIC. In this utility we have combined several of these techniques along with a useful assembly language routine for identifying control characters. The assembly language routine is stored in strings in the BASIC program and then read and poked into memory in a loop. This has the advantage over pokes in that the hex values can be seen and edited much easier rather than trying to decipher what a long list of poke statements mean.

Another feature of this utility is that the CATALOG command is issued directly as a monitor command from the assembly language program. If you think about it, how does one issue a catalog command from assembly language, or any monitor command for that matter? You can quickly see that while getting a catalog listing from BASIC is easy, doing the same thing from an assembly language routine is not. And so this routine also allows a method of executing monitor commands in a very efficient manner.

PROGRAM EXECUTION

To execute this program, simply type it in and type run. The program will then print out a catalog showing any control characters as "^A" for control-A etc. In this way any embedded control characters are easily identified. After the program executes, the DOS hooks are restored and this feature is eliminated. To keep this feature in effect after the program runs, delete line 2030, or simply run the program twice in succession. If this is done however, it will be necessary to re-boot to restore the system. Also, since all control characters are trapped in this manner, any disk commands in a program will not execute with this feature in effect, since DOS commands must be preceeded by a control-D character. In order to re-run the program after using it once, it must be re-loaded. This is the only way to enable the feature without an error, since running the program twice in succession causes a DOS error that

leaves the feature in effect. Thus to execute the program without leaving the trapping feature permanently in effect, some DOS command such as a LOAD must be executed between RUN's to re-set DOS pointers correctly. If anyone can identify why this is necessary and the appropriate DOS hook that is not being reset after the program executes, please write to The Apple Shoppe and we will print that information.

HOW IT WORKS

As can be seen in lines 0 to 120, the first part of the program sets up the assembly language routine as a series of string statements which are concatenated together to form a single string Y$ preceeded by "300: ". The effect of doing this is to mimic the monitor command of typing in a list of hex numbers and having the monitor store those values at each succeeding memory location from hex 300 on up. The end of the string is "N E88AG" which enables normal mode and gives the monitor "go" command to E88A which enables a running return to the current BASIC. This allows a way of automatically returning to BASIC from the monitor without keyboard intervention.

```
LIST 0,120

  0 DIM Y$(255),X1$(255),X2$(255
    ),X3$(255),X4$(255),X5$(255
    )
  1 X1$="48 29 7F C9 20 90 04 68 "

  2 X2$="4C BD 9E C9 0D F0 F8 C9 "

  3 X3$="15 F0 F4 C9 08 F0 F0 C9 "

  4 X4$="1B F0 EC A9 DE 20 F0 FD "

  5 X5$="68 69 40 4C BD 9E 00"
100 Y$="300: ":Y$( LEN(Y$)+1)=X1$
    :Y$( LEN(Y$)+1)=X2$:Y$( LEN(
    Y$)+1)=X3$
110 Y$( LEN(Y$)+1)=X4$:Y$( LEN(
    Y$)+1)=X5$
120 Y$( LEN(Y$)+1)=" N E88AG"
```

In the next line we complete the above operation by poking the above string into the input buffer at hex 200 and then telling the monitor that it has input by doing a call to -144. This passes control to the monitor, which will read the input buffer and execute the string of commands just as if we had typed it from the keyboard.

```
>
 LIST 130

 130 FOR I=1 TO LEN(Y$): POKE 511
     +I, ASC(Y$(I)): NEXT I: POKE
     72,0: CALL -144
```

The next section repeats the above scenario only this time we execute a catalog command from the monitor by placing in the monitor input buffer the address of the DOS catalog command. This allows automatic execution of the catalog command after we have reset the DOS output hooks to point to our assembly language routine at location 300. There the output characters will be checked for control characters and if found, the "^" symbol will be output along with the normal character represented by the control character. This allows identification of the embedded control character.

```
LIST 1500,3000
1500 Y$=""
2000 A= PEEK (-25340):B= PEEK (-
     25339)
2010 POKE -25340,0: POKE -25339,
     3: REM  DOS PRINT ROUTINE ADDRES
     S
2015 Y$="A56EG N E88AG"
2017 FOR I=1 TO LEN(Y$): POKE 511
     +I, ASC(Y$(I)): NEXT I
2018 POKE 72,0: REM  RESET STATUS REG
     .
2020 CALL -144: REM  FF70 ZMODE
2030 POKE -25340,A: POKE -25339,
     B
2035 REM
          A56E IS CATALOG FUNCTION IN
      DOS 3.2 WHILE E88A IS INTEGER B
     ASIC RUNNING RETURN. FOR APPLESO
     FT USE D823.
```

```
2036 REM
        SEE CALL-APPLE PAGE 6 JUNE
    1979 FOR REFERENCE.

2040 END

>
```

This completes the program. Note that the DOS output hooks are restored in line 2030 so that the printing of control characters is eliminated after the program runs. Remember not to try and run the program twice in succession without re-loading it first to restore all the DOS hooks, or this feature will remain in effect until the system is re-booted.

Looking in the Apple reference manual on page 169 will identify the location FF70 which is a call -144. This monitor location resets the monitor mode and falls into the routine that scans the input buffer for some more monitor commands. Since we stuffed in the BASIC return address onto the end of our monitor command list, we are assured of returning to BASIC after this call is made. This return to BASIC location is unique to integer basic and if using applesoft, must be replaced by D823 instead. Since the assembly languge routine is a bit hard to decipher within the BASIC listing, it is reproduced here. As can be seen, it is simply a series of compare statements that check for important control characters. If not a control character, the A register is pulled off the stack and a return is made by doing a jump to $9EBD in DOS. This location is supposed to call the correct print routine within DOS depending on the mode. Since we have replaced the current print address with hex 300, all DOS printing to the screen will pass first through locations 300 where we can check for control characters. Then it resumes its normal DOS output. For more information on the internals of DOS, see the DOS 3.2 memory map in the October 1979 issue of MICRO magazine and the August 1979 issue of MICRO magazine. The September issue of Applesauce, 1979 also has some mapping information for DOS 3.2. It is not known yet how much compatability is present between DOS 3.2.1 and DOS 3.3. Perhaps this can be the subject of a future article.

```
*
 300L

0300-   48          PHA
0301-   29 7F       AND   #$7F
0303-   C9 20       CMP   #$20
0305-   90 04       BCC   $030B
0307-   68          PLA
0308-   4C BD 9E    JMP   $9EBD
030B-   C9 0D       CMP   #$0D
030D-   F0 F8       BEQ   $0307
030F-   C9 15       CMP   #$15
0311-   F0 F4       BEQ   $0307
0313-   C9 08       CMP   #$08
0315-   F0 F0       BEQ   $0307
0317-   C9 1B       CMP   #$1B
0319-   F0 EC       BEQ   $0307
031B-   A9 DE       LDA   #$DE
031D-   20 F0 FD    JSR   $FDF0
0320-   68          PLA
0321-   69 40       ADC   #$40
0323-   4C BD 9E    JMP   $9EBD
0326-   00          BRK
*
```

For questions regarding this column please write the Utilities Workshop care of The Apple Shoppe magazine. We would welcome any article suggestions or information you may have found useful in dealing with the internals of your Apple computer.

GOLDCOAST COMPUTER RENTAL CLUB
10.00 YEARLY MEMBERSHIP FEE GETS YOU; ACCESS TO OVER (3000 PROGRAMS) PROFESSIONAL TOP QUALITY PROGRAMS TO RENT. EACH NEW MEMBER RECEIVES A TOP QUALITY DISKETTE FULL OF SELECTED PROGRAMS WORTH($50.00?).
BOX 142 ROUTE#1 BREMEN, KY 42325

PROGRAM REVIEW

BY
KIRK CHASE
PLACENTIA, CA.

HOUSE OF USHER

A chance for $100, a mystery to solve and monsters to battle is what you get when you play "House of Usher" by John and Patty Bell from Crystal Computer. The reward is real if you're the first to solve the mystery. This, of course is not very easy to accomplish. In fact, with all the creepy, crawly creatures, you may wonder if you will be able to survive the night!

This program is guaranteed to provide you with excitement as you go from room to room to room...to room. Pitfalls are placed in unlikely area. The game is made even more challenging by the fact you must keep up your stamina to avoid collapsing to your death from exhaustion. "House of Usher" is filled with objects that might aid or detour you away from solving your mystery.

The game is set up to be won in one of three ways. For the traveler who might have just happened upon the house, increasing your wealth is one way to win. For the adventurer, you may win by battlingthe multitudes of beasties. For the sleuth, and the way to win $100, is to solve the mystery. Any way you try is bound to be a challenge.

In fact, a fourth way to win may be added. And that is to survive your tour. The monsters that are found around the house are very good at what they do. With a few tricks, though, they only become a challenge to overcome. Also your stamina will decrease to zero before too long, and the quest for food soon becomes frantic. It is very easy to die and not so easy to live in the "House of Usher", one of a family of adventure games from Crystal Computer. See it today at your local Computer Store. It's well worth the money!

THE WIZARD AND THE PRINCESS

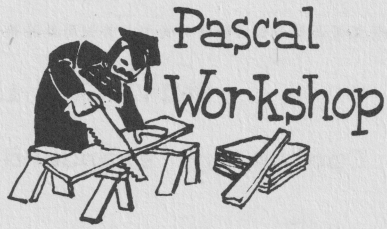
Battle your way passed a multitude of obstacles to the castle of an evil wizard. Your quest is to rescue a beautiful princess from the clutches of this evil wizard. This is the theme of Hi-Res Adventure #2 by Ken and Roberta Williams of On-Line Systems. Ken and Roberta have given a wonderful new meaning to adventure games with this, their latest new adventure disk.

With the help of high-resolution graphics, you tour many different terrains on your quest. You start off with a pocket knife, a loaf of bread, a flask of water and a blanket. But you soon discover this is not enough to make your trek through the beautiful graphics. For this collection is little in the way to save the princess against a wizard.

This adventure is extremely challengin, and it will require tremendous brain work and imagination to make any progress. This program is difficult and so you are encouraged to save the game at certain points of progress on the floppy disk, to return later, more refreshed. Like many adventures, it is easy to die and not very easy to stay alive. Saving the game along the way saves you time in bringing yourself up to date to where you were in the adventure.

Running "The Wizard and the Princess" is pretty straight forward as it has only two mazes. The program also has it's share of exotic creatures and situations. Snakes stop your progress in the beginning. A giant, lion and creeky bridge are more obstacles in your path. If you come to a point where you are stopped for an unusually long time with no ideas of how to surpass the obstical, check with your local computer store or call On-Line Systems for a hint. Ken will be glad to ease your suffering!

"The Wizard and the Princess" is a wonderful software disk full of high resolution graphics. You become an adventurer with the computer passing great obstacles on your way to save the princess. The program requires days or even weeks to complete, and all-in-all, it is a great game from Ken Williams and On-Line Systems. See it today at your favorite Computer Store.



16 SECTOR DISK COPY

This month's Pascal Workshop features a cure for a very troubling Apple got-cha. Namely that Apple FORTRAN cannot be copied and backed up. This program can solve that problem if the user so chooses since it copies 16 sector pascal disks. We appreciate all the readers who submitted solutions to this problem and we are sorry we cannot acknowledge all of you. The author of this particular solution is Allen W. Todd from Albany Oregon and we appreciate his contribution.

Copy 16 is a program written in U.C.S.D. Pascal and runs on the Apple II with either Pascal or Fortran operating systems. It enables the user to copy entire 16 sector diskettes in either of two modes. The first mode is the same as that used by the Pascal file management system (Filer). The copy is made by reading 56 blocks (28K) from the source diskette into a buffer, then writing this buffer to the destination diskette. This process is repeated five times to copy the entire 280 blocks. If diskettes with other than 280 blocks are to be copied, a few constants in the source code may be changed and the program re-compiled.

The second mode of copying is slower than the first, but allows for error checking on each block. In the error checking mode, all bad blocks will not be transfered but rather will be replaced on the destination diskette by blocks of zeros. Copy 16 will copy all standard Pascal, Fortran or DOS 3.3 diskettes currently available for the Apple II. A minimum of two disk drives are required.

```
{**************************************************************************
*                                                                         *
*              UNIVERSAL 16 SECTOR DISKETTE COPY PROGRAM                  *
*                                                                         *
*     Copies any standard Apple 16 sectored Pascal, Fortran, or DOS       *
*                              diskette                                   *
*                                                                         *
*                                 by                                      *
*                            Allen W. Todd                                *
*                     Todd Microcomputer Consultants                      *
*                        2711 N.W. Gibson Hill Rd.                        *
*                          Albany, Oregon  97321                          *
*                             (503) 926-2986                              *
*                                                                         *
**************************************************************************}

PROGRAM UNIVERSALCOPY;
{$I-,R-}

CONST
   buflen = 28672;
   numblocks = 56;

TYPE
   byte = 0..255;

VAR
   tovol, fromvol: INTEGER;
   eosclear, eolnclear, home: PACKED ARRAY[0..1] OF CHAR;
   toname, fromname: string;
   ch: CHAR;

PROCEDURE getcrtinfo;  {Terminal Independant Screen I/O}
TYPE
   bitorbyte = PACKED RECORD CASE BOOLEAN OF
                  true: (bbyte: byte);
                  false: (bit: PACKED ARRAY[0..15] OF BOOLEAN);
               END;
   syscomrec =  PACKED ARRAY[0..511] OF CHAR;
   syscomptr = ^syscomrec;
VAR
   syscom: syscomptr;
   trix: RECORD CASE BOOLEAN OF
            false: (address: INTEGER);
            true:  (pointer: ^syscomptr);
         END;
   prefixed: bitorbyte;
   leadin: CHAR;

BEGIN
   trix.address:= 248; {Pascal system pointer to syscom ($F8)}
   syscom:= trix.pointer^;
   leadin:= syscom^[62];
   eosclear[1]:= syscom^[64];
   eolnclear[1]:= syscom^[65];
   home[1]:= syscom^[63];
   WITH prefixed DO
      BEGIN
```

```
          bbyte:= ord(syscom^[72]);
          IF bit[6]
             THEN home[0]:= leadin
             ELSE home[0]:= chr(0);
          IF bit[3]
             THEN eosclear[0]:= leadin
             ELSE eosclear[0]:= chr(0);
          IF bit[2]
             THEN eolnclear[0]:= leadin
             ELSE eolnclear[0]:= chr(0);
       END; {with}
END; {getcrtinfo}
FUNCTION badio: BOOLEAN; {I/O error routine}

VAR
   iocode: 0..16;

BEGIN
   iocode:= IORESULT;
   badio:= false;
   IF iocode > 0 THEN
      BEGIN
         badio:= true;
         writeln;
         write ('I/O ERROR:  ');
         CASE iocode OF
            1: writeln ('Bad Block, Parity Read Error');
            2: writeln ('Bad Unit Number');
            3: writeln ('Bad Mode, Illegal Operation');
            4: writeln ('Undefined Hardware Error');
            5: writeln ('Lost Unit, Unit is no longer on line');
            6: writeln ('Lost File, File is no longer in Directory');
            7: writeln ('Bad Title, Illegal File Name');
            8: writeln ('No Room on Disk, Insufficient space');
            9: writeln ('No Unit, No such volume on line');
            10: writeln ('No File, No such file on volume');
            11: writeln ('Duplicate File');
            12: writeln ('File not Closed, Attempt to open an open file');
            13: writeln ('Not Open, Attempt to access a closed file');
            14: writeln ('Bad Format, Error in reading real or integer');
            15: writeln ('Ring Buffer Overflow');
            16: writeln ('Write Protected');
         END; {Cases}
         writeln;
         writeln ('PRESS (SPACEBAR) TO CONTINUE');
         read (keyboard,ch);
      END; {If}
END; {Badio}
PROCEDURE abort;
BEGIN
   write (home,eosclear);
   gotoxy (0,10);
   writeln ('Universal Copy Program Aborted.........');
   exit (program);
END; {abort}

PROCEDURE copy; {copies diskette with checking for bad blocks}
```

```
VAR
   buffer: PACKED ARRAY[0..buflen] OF byte;
   bnum, base, index, i, badnum: INTEGER;

BEGIN
   bnum:= 0;
   badnum:= 0;
   gotoxy (0,15);
   write ('Copying');
   REPEAT
      index:= 0;
      base:= bnum;
      REPEAT
         unitread (fromvol,buffer[index],512,bnum);
         IF ioresult = 0
            THEN write ('.')     {good block}
            ELSE
               BEGIN             {bad block, zero it on copy diskette}
                  write ('[',bnum,']');
                  FOR i:= index TO index + 511 DO buffer[i]:= 0;
                  badnum:= badnum+1;
               END;
         bnum:= bnum + 1;
         index:= index + 512;
      UNTIL (bnum > 279) OR (bnum MOD numblocks = 0);
      unitwrite (tovol,buffer,buflen,base);
      IF badio THEN abort;
   UNTIL bnum > 279;
   gotoxy (0,20);
   writeln (fromname,' ------> ',toname);
   writeln (badnum,' Bad Blocks Encountered');
END; {copy}

PROCEDURE copyfast; {copies diskette quickly without error checking}

VAR
   buffer: PACKED ARRAY[0..buflen] OF byte;
   bnum: INTEGER;

BEGIN
   bnum:= 0;
   REPEAT
      gotoxy (0,15);
      write (eolnclear,'Reading blocks ',bnum,' to ',bnum + numblocks - 1);
      unitread (fromvol,buffer,buflen,bnum);
      IF badio THEN abort;
      gotoxy (0,15);
      write (eolnclear,'Writing blocks ',bnum,' to ',bnum + numblocks - 1);
      unitwrite (tovol,buffer,buflen,bnum);
      IF badio THEN abort;
      bnum:= bnum + numblocks;
   UNTIL bnum > 279;
   gotoxy (0,20);
   writeln (fromname,' ------> ',toname);
END; {copyfast}

FUNCTION getdirectory (dunit: INTEGER; VAR vname: string): BOOLEAN;

{check for formatted diskette, get diskette type, and volume name}
```

```
TYPE
   vid = string[7];
   tid = string[15];
   daterec = PACKED RECORD
                mo:    1..12;
                day:   1..31;
                year:  0..99;
             END;
   filekind = (vol, badfile, code, text, info, data, graf, foto, secure);
   direntry = RECORD
                dfirstblock: INTEGER;
                dlastblock:  INTEGER;
                CASE dfkind: filekind OF
                   vol, secure:  (dvid:       vid;
                                  deovblk:    INTEGER;
                                  dfilenum:   INTEGER;
                                  ddummy:     INTEGER;
                                  dlastboot: daterec);
                   badfile, code, text, info, data, graf, foto:
                                 (dtid:       tid;
                                  dlastbyte: 1..512;
                                  daccess:   daterec)
             END;
   directory =  ARRAY[0..77] OF direntry;

VAR
   dirptr: ^directory;
   heap: ^INTEGER;
   s: string;

BEGIN
   getdirectory:= false;
   str (dunit,s);
   mark (heap);
   new (dirptr);
   unitread (dunit,dirptr^,sizeof(directory),2); {directory begins on block 2}
   IF NOT badio
      THEN WITH dirptr^[0] DO
         BEGIN
            IF (dfkind IN [vol,secure]) AND (deovblk=280)
               THEN vname:= dvid
               ELSE vname:= concat ('DOS VOLUME #',s);
            getdirectory:= true;
         END; {with}
   release (heap);
END; {getdirectory}

BEGIN  {Universal Copy}
   getcrtinfo;
   REPEAT
      write (home,eosclear);
      writeln ('Universal 16 Sector Diskette Copy Program');
      REPEAT
         gotoxy (0,5);
         write (eosclear,'Copy Which Disk Volume?  ');
         readln (fromvol);
      UNTIL (NOT badio) AND (fromvol IN [4,5,9..12]);
      REPEAT
         gotoxy (0,7);
```

```
      write (eosclear,'To Which Disk Volume?  ');
      readln (tovol);
   UNTIL (NOT badio) AND (tovol IN [4,5,9..12]) AND (fromvol()tovol);
   IF NOT getdirectory (tovol,toname) THEN abort;
   IF NOT getdirectory (fromvol,fromname) THEN abort;
   gotoxy (0,10);
   write ('Destroy ',toname,'?  ');
   read (ch);
   IF (ch IN ['Y','y']) THEN
      BEGIN
         gotoxy (0,12);
         write ('Check for bad diskette blocks?  ');
         read (ch);
         IF (ch IN ['Y','y'])
            THEN copy
            ELSE copyfast;
      END;
   gotoxy (0,23);
   write ('More to Copy?  ');
   read (ch);
UNTIL NOT (ch IN ['Y','y']);
write (home,eosclear);
gotoxy (0,10);
writeln ('Universal Copy Ended..........');
END.
```

STRING SEARCH UTILITY

Several readers have written in requesting help with such programs as "File Cabinet" particularly in the area of string matching. These requests have given us an idea for a useful utility that can be inserted into any data base application that involves matching a search string with items in the data base. This month we present such a utility which can be added as a subroutine to such programs as "File Cabinet" to increase their usefulness.

It we think about what we would like in such a utility first, then we can define what the subroutine must do. Here is a list of desirable objectives.

REQUIREMENTS

1. Search target string for the object string.

2. Continue search for each occourance of the object string in the target string.

3. Allow "don't care" positions in the object string.

4. Be flexable enough and straight forward so as to be adaptable to a variety of uses.

Now that we have identified several requirements, we must translate those requirements into more specific instructions that will help us figure out how to write the program.

The first requirement simply says given a target string, find the first occourance of the object

string. This implies that we must search the target string from beginning to end to see if the first letter of our object string is contained anywhere in the target string. If not, then their can be no match. If it is, then their might be a match if the remianing letters agree. Here is the first part of our program. To use, we simply let TARGT$= "string to be searched" and the OBJ$="string to search for" prior to calling this subroutine.

```
]LIST 0,230
 10  REM  MAIN PROGRAM

 20 TARGT$ = "NOW IS THE TIME FOR
      ALL GOOD MEN TO COME TO THEI
      R COUNTRY'S AID!"
 30 OBJ$ = "T#"
 40  GOSUB 100
 50  END : REM  OF MAIN
100  REM
           STRING SEARCH UTILITY
110  REM
           STRING MAGIC WORKSHOP
           APPLE SHOPPE VOL1 NO.8
           (C) COMPU-TUTOR 1980
200  REM
           BEGIN MAIN;

210  REM
           VAR;

220  REM
           TARGT$=STRING TO BE
           SEARCHED.

230  REM
           OBJ$=MATCH STRING
           '#'=DON'T CARE.
```

Lines 10 through 50 give an example of how the subroutine would be called from our main program. Note how the target and object strings are set-up and the utility called in line 40. These lines would be adapted to the program in which you are using the utility. The actual subroutine begins at line 100. Lets look at the first portion of the subroutine now.

```
]LIST 240,290

240 ZI = 1:ZW = 1: REM  COUNTERS
250 NOGO = 1: REM  FLAG
260  HOME : HTAB 3: INVERSE : PRINT
     "STRING SEARCH UTILITY": NORMAL
     : VTAB 3
270  PRINT "SEARCH STRING: ";: INVERSE
     : PRINT OBJ$: NORMAL : VTAB
     8
```

Here we simply print the target and object strings and set-up our string pointers ZI and ZW. The NOGO variable is used to pass back information on whether or not a match has been found. Now we are ready to being our search.

```
]LIST 300,339

300  REM
          BEGIN SEARCH;

310  FOR ZZ = 1 TO  LEN (TARGT$)
320  IF  MID$ (OBJ$,1,1) =  MID$
     (TARGT$,ZZ,1) THEN  GOSUB 50
     0
330  IF NOGO = 0 THEN NOGO = 1: GOTO
     340
335  NEXT ZZ
338  GOTO 390
```

The key to our search is line 310 and 320. Here we set up a loop for the entire length of the target string. Then we test each character in the target string against the first character in the object string. If we have a match then we GOSUB 500 to see if the remaining characters match. If not, then we continue until the length of the target string is exhausted. Obviously if the first character in the object string never matches anything in the target string, then there never will be a match. We assume here that the "dont't care" character is never the first character in the object string. In line 338 we branch to the exit portion at line 390 where we return to the main program. Note that NOGO is

initialized to 1 so as to prevent branching in line 330 if we did not GOSUB 500. Often when writing a program it is easy to forget that the first time through a loop variables must be initialized properly to prevent triggering unwanted branching before the main event. Lets assume that in line 320 we did branch to our search subroutine and as a result NOGO was reset to 0 indicating we have found out object string.

```
]LIST 340,499

340  REM
         MATCH FOUND

350  PRINT : PRINT "STRING FOUND:
     ": PRINT : PRINT
360  PRINT  MID$ (TARGT$,1,(ZZ -
     1));
370  INVERSE : PRINT  MID$ (TARGT
     $,ZZ, LEN (OBJ$));
380  NORMAL : PRINT  MID$ (TARGT$
     ,ZZ +  LEN (OBJ$), LEN (TARG
     T$) - (ZZ +  LEN (OBJ$)))
382  PRINT : PRINT "    PRESS KEY
      TO CONT...";: GET KEY$: PRINT

385  NEXT ZZ
390  PRINT : PRINT "    THAT'S AL
     L FOLKS!": RETURN
395  END : REM  OF UTILITY
```

Line 380 identifies the portion of the target string that matched the object string. This string could be saved in another return variable if desired to return it to the main program. Line 382 simply allows us to pause before continuing our search for any other occourances of our object string in the target string. This line could be removed when using the utility in an on-going program. Line 390 then returns us to the main program. Now finally we come to the key seach subroutine that must determine if the remaining object string including the "don't care" characters match up with the initial match position in the target string as pointed to by the ZZ pointer. Note the use of the second ZW pointer that tracks the object string length.

```
]LIST 500,600

500  REM
         CHECK IF MATCH

510  FOR ZW = 1 TO  LEN (OBJ$)
520  IF  MID$ (OBJ$,ZW,1) = "#" THEN
     540
530  IF  MID$ (OBJ$,ZW,1) < > MID$
     (TARGT$,ZZ + ZW - 1,1) THEN
     NOGO = 1: RETURN
540  NEXT ZW
550 NOGO = 0: RETURN

]PR#0
```

To handle the "don't care" positions, we simply skip over any character in the target string at those positions. as shown in line 520. Line 530 then determines if the next character in the object string matches with the corresponding character in the target string. If so, then we continue with line 540 and NEXT ZW. If not, then our match has failed at that point and our NOGO flag is set to 1 indicating a failure and we return to line 330. Note that line 550 is necessary to handle the case where we reach the end of the object string without finding a match. Little "got-cha's" like that is why 95% of any program is quickly written, while the remaining 5% of the program often takes as long as the first 95%. Because debugging and identifying all possible program conditions takes a lot of time and careful testing.

Type this utility in and try it a few times just the way it is written. Once you understand how it works, then you can insert it into your own programs for finding data in record fields or any other application where matching is necessary. If you have an idea for the "STRING MAGIC" WORKSHOP please send it to the Apple Shoppe. Until next time, try some more STRING MAGIC in your BASIC programs!

THE BOOK IS COMING!

Computerland
16720 Hawthorne Blvd.
Lawndale, CA. 90260

CRYSTALWARE

Hires graphics - Crystalsonics - sophistication...Crystal is what is happening in fantasy and science fiction games! Tired of text games and wordy manuals? Have you been looking for games which are different each time you play? We offer you the results of three years of testing and research; what we believe to be the finest games available anywhere for the Apple Computer.

HOUSE OF USHER — Wander through a haunted house. Rooms and scenery in 15 color lo-res graphics. We offer a $100.00 prize to the first person to solve the mystery. Over 200 monsters, objects, and perils. **$24.95**

GALACTIC QUEST — Crystalsonics - hires graphics - the ultimate space adventure. Vegan warships attack and fire in real time simulation. Land on and trade with over 64 star systems in 3 galaxies. Allow 6-12 hours for play. **$24.95**

SUMER — Travel back through time to ancient Sumeria in the middle east. You are given 10 years as king to restore this kingdom to prosperity. Plant, war, consult the astrologers - very hard to beat! **$19.95**

LASAR WARS — Crystalsonics - hires graphics - protect the planet earth from a full scale alien invasion. Over three types of invading craft and hundreds of approach simulations. The game speaks for itself. **$29.95**

WORLD WAR III — Crystalsonics - hires graphics - for you war game freaks, this is it! Iran and Iraq - nuclear missiles - hires 3 sceeen battlefield - demolition squads - tanks - strategy. Custom designed for two arm chair generals. Save the world from nuclear holocaust! **$29.95**

BENEATH THE PYRAMID — Crystalsonics - hires graphics - brand new! Explore the pyramids and miles of winding secret tunnels beneath them. Enter at the Sphinx and find the hidden treasure chamber. All in hires with very aggressive monsters and many many perils. To win you must find the golden cat and your way out!!! **$29.95**

for Apple 48K with disc

Please send all correspondence to P.O. Box 206 San Jose, Calif. 95113

THE APPLE SHOPPE
P.O. BOX 22026
SAN DIEGO, CA 92122

BULK RATE
U.S. POSTAGE
PAID
San Diego, Ca.
Permit No. 2155